Stealth Distributed Hash Table: Unleashing the Real Potential of Peer-to-Peer

Andrew Brampton, Andrew MacQuire, Idris A. Rai
Nicholas J. P. Race and Laurent Mathy
Computing Department
Lancaster University
Lancaster, LA1 4WA, UK
{brampton,macquire,rai,race,laurent}@comp.lancs.ac.uk

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*;
C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms, Performance, Experimentation

Keywords

Distributed Hash Tables, Peer-to-Peer, Stealth DHT

1. INTRODUCTION

Distributed Hash Tables (DHTs) have been shown to be a promising form of decentralised structured peer-to-peer networking, offering substantial scalability and resilience. Unsurprisingly, there exist numerous DHT systems [11][13][16]. Primarily, DHTs serve as an object location service that can be used as a substrate for multiple large-scale distributed applications such as storage [4][9], multicast [3][2] and load balancing systems [8][6].

Historically, the approach to DHT design has involved connecting an overlay of *homogeneous* and *autonomous* nodes together. The homogeneity arises from the fact that nodes are assumed to have similar properties such as network bandwidth and network stability. On the other hand, the autonomicity of nodes arises in the sense that any node may join or leave the network, and perform any operation supported by the DHT such as *routing messages* or handling object references (*keys*) as they wish.

Realistically, measurement studies have shown that nodes are not homogeneous [14] and nodes are likely to be continually joining and leaving the network in an unpredictable fashion, a situation commonly referred to as *churn*. Many DHT systems will simply break down under high churn [12], unfortunately, severe levels of churn are likely, especially for networks with mobile nodes [7][10].

Another issue is that of trust among users: traditional content-delivery models separate clients and servers, unlike DHT-based content systems where the end-users play a role in the content location and delivery. It is unwise to allow all nodes the same privileges on the network due to the potential for abuse [15][5].

We propose a *Stealth Distributed Hash Table* (Stealth DHT) algorithm, which aims to preserve the advantages of DHT-based systems, while offering greater security and control at the DHT level. A Stealth DHT makes a subset of nodes effectively "invisible" to the routing decisions on the network. As a result, invisible nodes will never receive any queries and therefore cannot intercept nor reply to them.

2. STEALTH DHT

There are two types of nodes in a Stealth DHT, namely *service nodes* and *stealth nodes*. Service nodes can execute all operations supported by the DHT, while stealth nodes are prevented from storing keys and forwarding data. In a commercial context, for example, service nodes could be highly stable and capable machines owned by a content service provider, and they are assumed to be trusted; on the other hand, stealth nodes would be autonomous devices owned by end-users who request service(s) from the provider. Note however, that the assignment of role (service or stealth) to nodes is application dependent and in no way prescribed or constrained by the Stealth DHT itself.

To this end, the routing state of all nodes in the Stealth DHT contains entries for only service nodes. Consequently, any node (and in particular stealth nodes) can only send messages to service nodes. The service nodes then forward the message (only via other service nodes) to the destination, which will also be a service node. In other words, stealth nodes cannot communicate with one another, nor will service nodes communicate with stealth nodes for any purpose other then replying directly to requests issued by stealth nodes. Consequently, stealth nodes are invisible to each other, and when "quiet", their presence is invisible to the service nodes.

To achieve the differentiation between service and stealth nodes, stealth nodes make use of a lightweight join mechanism. This mechanism prevents the stealth node announcing their presence to the network, thus keeping them out of the other nodes' routing tables. Intuitively, when a stealth node joins, no routing updates are required, and when a stealth node leaves, no routing entries become stale.

As stealth nodes do not appear in the service nodes' routing tables, the stealth nodes do not receive updates to their own tables. This can lead to a stealth node having an increasingly stale routing table. There are three possible low-cost mechanisms to tackle this, piggybacking routing infor-

CoNEXT'05, October 24–27, 2005, Toulouse, France.
ACM 1-59593-035-3/05/0010.

mation on replies to stealth nodes, periodically polling for routing state, and rejoining the node to the network.

3. PERFORMANCE EVALUATION

We implemented both Pastry [13] (i.e. a generic DHT) and our Stealth DHT in our own discrete-event packet-level simulator. The underlying network consisted of 1000 routers on a transit-stub network (4% transit nodes), generated with GT-ITM [1]. Peers were attached to the physical network in a random fashion.

Simulations were run with a realistic workload in which randomly selected nodes performed *put* and *get* operations on a set of 1,000,000 keys. These simulations were run for both Pastry and a Stealth DHT, with and without churn in the network.

Regardless of churn Stealth DHTs outperform Pastry in many standard DHT measurements, such as average hop count, relative delay penalty (often referred to as stretch), join overhead and load balancing. We also found that in a Stealth DHT increasing the number of stealth nodes had no significant impact on these metrics.

Without churn, the cost of using a Stealth DHT was increased link stress for a small percentage of the network compared to Pastry, as well as higher average load (the number handled message) for the service nodes. Under churn, however, generic DHTs generate a large number of messages to detect and repair the failures. Overall, under churn the Stealth DHT was found to have a lower average and maximum link stress, as well as reduced average load per node. We attribute this to the lightweight joining protocol for stealth nodes, and the lack of repair messages required when stealth nodes leave the DHT.

4. CONCLUSION

Most of the research in DHT-based systems has focused on performance issues. However, a generic DHT design does not generally address issues such as heterogeneous nodes in a network and security.

We proposed a new DHT paradigm called Stealth DHT that is capable of tackling these problems. In particular, a Stealth DHT simplifies the DHT operations for stealth nodes, which leads to improved performance. In addition, by preventing stealth nodes from forwarding messages, the service nodes can actually restrict the content stored in the DHT and enforce control over network operations, thus improving security.

Stealth DHT could support commercial applications where the distributed and resilient aspect of the DHT is required, but where the content stored in the network needs to be limited to licence materials. Mobility is another area where Stealth DHTs could be used, by restricting mobile devices to stealth nodes their churn would not adversely affect the network.

Consequently, Stealth DHTs are poised to improve support for existing DHT applications in real world communication environments as well as enabling the commercial development of the technology.

5. REFERENCES

[1] K. Calvert and E. Zegura. Georgiatech internetwork topology models. http://www.cc.gatech.edu/projects/gtitm.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in a cooperative environment. In *Proc. of ACM SOSP*, October 2003.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications*, 20(8), 2002.

[4] P. Druschel and A. Rowstron. A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS VIII)*, pages 75–80, May 2001.

[5] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. In *Proc. of ACM SIGMETRICS*, pages 38–49, June 2005.

[6] P. B. Godfrey and I. Stoica. Heterogeneity and load balance in distributed hash tables. In *Proc. of IEEE INFOCOM*, March 2005.

[7] H.-C. Hsiao and C.-T. King. Mobility churn in DHTs. In *Proc. of the 1st International Workshop on Mobility in Peer-to-Peer Systems (MPPS'05) in conjunction with the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 799–805, June 2005.

[8] D. R. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 36–43, June 2004.

[9] J. Kubiatowicz. Oceanstore: An architecture for global-scalable persistent storage. In *Proc. of the ASPLOS 2000*, November 2000.

[10] H. Pucha, S. M. Das, and Y. C. Hu. How to implement DHTs in mobile ad hoc networks? In *Proc. of the 10th ACM International Conference on Mobile Computing and Network (MobiCom 2004)*, September 2004.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, August 2001.

[12] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proc. of the USENIX Annual Technical Conference*, June 2004.

[13] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.

[14] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN*, 2002.

[15] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. of IPTPS*, March 2002.

[16] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, August 2001.